

CODING CLUB

EPITECH LYON



CPool Day -1

VERSION 1.0.1

Prémices

L'objectif de ce Coding club est de vous familiariser avec le langage C et de vous offrir un avant goût de la forme mais aussi de la rigueur de programmation qui vous sera demandé en octobre prochain

Et pour que votre périple à travers la journée -1 de votre piscine de C se passe à merveilles il y a quelques règles à respecter !

Règles :

- Chaque exercice a été pensé afin d'être réalisable avec pour **seul syscall la fonction write()** de la librairie **unistd** . Toute fonction externe utilisée dans votre exercice [*exemple printf(), puts()*] déclenchera une erreur dans notre moulinette et refusera catégoriquement la validation de votre exercice aussi juste soit-il.
- Vous devez organiser votre travail dans une arborescence de dossiers cohérente afin de faciliter le fonctionnement de la moulinette. Au début de chaque exercice le nom du fichier à rendre ainsi que le dossier de rendu vous seront communiqués. Exemple d'arborescence:

```
Coding_Club/ |  
              + Hello/  
              |   |- hello.c  
              + Digits/  
              .   |- digits.c
```

- Si vous avez une question, demandez à votre voisin de droite. Si malgré tous les efforts de votre cher voisin le problème persiste, rabattez vous par défaut sur votre voisin de gauche. Si vous êtes en bout de table et donc dans l'impossibilité d'avoir un voisin de droite ou de gauche il vous reste toujours celui de devant !
- Si malgré cela une question reste en suspens ou un problème vous ennuie un peu trop, n'hésitez pas à appeler un cobra ils sont là pour ça !
- Tout au long de ces exercices nous vous demanderons de réaliser des **fonctions**. Cela signifie que notre moulinette amènera son propre main pour tester votre travail. Si par hasard vous laissez votre fonction main() il se peut que cela ne marche pas et il se peut que cela soit de votre faute !


 Vous ne savez pas ce qu'est une fonction main ? Google it ;)

Moulinette

Tout au long de cette journée vous aurez accès à un script python de correction vous permettant de savoir si votre exercice est correct ou non.

Le script affichera alors une batterie de tests pour l'exercice sur le terminal sous la forme suivante :

[Nom du test]: PASSED ou FAILED

 PASSED signifie que vous avez validé le test.
FAILED signifie tout logiquement l'inverse.


Ce script, que nous appellerons moulinette, est un programme qui va venir corriger votre programme.

Il a été réalisé avec soin et ne peut pas avoir tort. Si un exercice fonctionne chez vous mais que la moulinette dit le contraire essayez d'imaginer tous les cas possibles en suivant les noms de tests pour deviner. (On fait même des devinettes dans le Coding club trop cool !)

Si vraiment vous avez l'impression que la moulinette se trompe. (*nb: Hautement improbable*)

Ou bien que malgré **toute votre réflexion** un test reste un mystère absolue pour vous n'hésitez pas à appeler un Cobra ils sont là pour ça :)

Pour utiliser cette fameuse moulinette il vous suffit de taper "mouli" dans votre terminal.

 **Attention:** un code vous sera demandé pour la lancer, ces codes seront distribués à intervalle régulier par les cobras pour éviter tout abus.

Prêt ?

Si tout ceci est clair comme de l'eau de roche et bien je crois qu'il est grand tout pour vous de commencer votre avant premier jour de piscine !

À vos marques... prêts... **Codez !**

Hello

Fichier de rendu: `hello/hello.c`

Sujet :

L'objectif de cet exercice est d'écrire une fonction qui affichera "Hello World!" suivi d'un retour à la ligne dans terminal.

La fonction devra être prototypée de la façon suivante :

```
void hello(void);
```

Exemple :

Un appel de votre fonction avec les arguments suivants devra afficher ces résultats :

```
hello();
```

```
$> hello
```

Comment faire pour afficher un retour à la ligne déjà ?

Digits

Fichier de rendu: `digits/digits.c`

Sujet :

Écrivez une fonction qui affiche dans le terminal tous les chiffres dans l'ordre croissant suivi d'un retour à la ligne.

La fonction devra être prototypée de la façon suivante :

```
void digits(void);
```

Exemple :

Un appel de votre fonction avec les arguments suivants devra afficher ces résultats :

```
digits();
```

```
$> 0123456789
```

Revalpha

Fichier de rendu: `revalpha/revalpha.c`

Sujet :

Écrivez une fonction qui affiche dans le terminal l'alphabet à l'envers une lettre sur deux en majuscule suivit d'un retour à la ligne

La fonction devra être prototypée de la façon suivante :

```
void revalpha(void);
```

Exemple :

Un appel de votre fonction avec les arguments suivants devra afficher ces résultats :

```
revalpha();
```

```
$> zYxWvUtSrQpOnMlKjIhGfEdCbA
```

Alandroit

Fichier de rendu: `alandroit/alandroit.c`

Sujet :

Écrivez une fonction qui affiche dans le terminal la chaîne de caractères passée en paramètre de la fonction.

La fonction devra être prototypée de la façon suivante :

```
void alandroit(char *str);
```

Exemple :

Un appel de votre fonction avec les arguments suivants devra afficher ces résultats :

```
alandroit("EPITECH");
```

```
$> EPITECH
```

```
alandroit("BEEP BOOP");
```

```
$> BEEP BOOP
```

Alanver

Fichier de rendu: `alanver/alanver.c`

Sujet :

Écrivez une fonction qui affiche à l'envers dans le terminal la chaîne de caractères passée en paramètre de la fonction.

La fonction devra être prototypée de la façon suivante :

```
void alanver(char *str);
```

Exemple :

Un appel de votre fonction avec les arguments suivants devra afficher ces résultats :

```
alanver("EPITECH");
```

```
$> HCETIPE
```

```
alanver("BEEP BOOP");
```

```
$> P00B PEEB
```

```
alanver("bob");
```

```
$> bob
```

Humm... Il faudrait sûrement écrire la chaîne de caractères en partant de la fin, mais comment savoir où est la fin ?

mY_sTr

Fichier de rendu: mY_sTr/mY_sTr.c

Sujet :

Écrivez une fonction qui, prenant pour paramètre une chaîne de caractères va inverser la casse de chacune de ses lettres puis l'afficher dans le terminal.

La fonction devra être prototypée de la façon suivante :

```
void mY_sTr(char *str);
```

Exemple :

Un appel de votre fonction avec les arguments suivants devra afficher ces résultats :

```
mY_sTr("Vive le Coding Club 2022");
```

```
$> vIVE LE cODING cLUB 2022
```

```
mY_sTr("1998");
```

```
$> 1998
```

```
mY_sTr("POURQUOI JE CRIE ?");
```

```
$> pourquoi je crie ?
```

Suspension

Fichier de rendu: `suspension/suspension.c`

Sujet :

Écrivez une fonction qui renvoie le nombre de caractères représentés par la variable `'to_find'` trouvés dans la chaîne de caractères passée en paramètre

La fonction devra être prototypée de la façon suivante :

```
int suspension(char *str, char to_find);
```

Exemple :

Un appel de votre fonction avec les arguments suivants devra afficher ces résultats :

```
printf("%d\n",suspension("EPITECH", 'E'));
```

```
$> 2
```

```
printf("%d\n",suspension("###3.##.###...###4#", '.'));
```

```
$> 5
```

Nous utilisons la fonction `printf` dans notre exemple afin d'illustrer une façon d'afficher la valeur de retour d'une fonction. Vous êtes bien évidemment invités à l'utiliser dans vos tests

:danger ⚠ Bien que `printf` soit utilisable à tout moment pour tester vos fonctions, `printf` reste bien évidemment interdit et sera flag par la moulinette.

Search_n_replace

Fichier de rendu: `search_n_replace/search_n_replace.c`

Sujet :

Écrivez une fonction qui, prenant 3 paramètres, va venir remplacer dans la chaîne de caractères paramètre 1 (str) le caractère paramètre 2 (a) par le caractère paramètre 3 (b) puis l'affiche dans le terminal suivi d'un retour à la ligne.

La fonction devra être prototypée de la façon suivante :

```
void search_n_replace(char *str, char a, char b);
```

Exemple :

Un appel de votre fonction avec les arguments suivants devra afficher ces résultats :

```
search_n_replace("EPITECH", 'E', '%');
```

```
$> %PIT%CH
```

```
search_n_replace("0w0", '0', 'U')
```

```
$> UwU
```

Repeat_alpha

Fichier de rendu: `repeat_alpha/repeat_alpha.c`

Sujet :

Pour cet exercice je vous laisse deviner le sujet par vous même !

Vous verrez c'est pas sorcier !

La fonction devra être prototypée de la façon suivante :

```
void repeat_alpha(char *str);
```

Exemple :

Un appel de votre fonction avec les arguments suivants devra afficher ces résultats :

```
repeat_alpha("abc");
```

```
$> abbccc
```

```
repeat_alpha("Alex");
```

```
$> Allllllllllllleeeeexxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

```
repeat_alpha("abacadaba 42!");
```

```
$> abbaccaddddabba 42!
```

Si vous n'arrivez vraiment pas à trouver une corrélation entre les exemples et après avoir bien réfléchi, vous pouvez appeler un cobra (On oublie pas aussi la règle avec les voisins ;p)

HiddenTF

Fichier de rendu: `hidentf/hidentf.c`

Sujet :

Écrivez une fonction qui renvoie 1 si toutes les lettres de `s1` se cachent dans `s2` et 0 dans le cas inverse.

La fonction devra être prototypée de la façon suivante :

```
int hidentf(char *s1, char *s2);
```

Exemple :

Un appel de votre fonction avec les arguments suivants devra afficher ces résultats :

```
printf("%d",hidentf("abc", "abcdefghjikqsdblalbla"));
```

```
$> 1
```

```
printf("%d",hidentf("fgex", "tyf34gdgf;'ektufjhgdgex.;.;rtjynur6"));
```

```
$> 1
```

```
printf("%d",hidentf("azbc", "btarc"));
```

```
$> 0
```

HiddenTF_order

Fichier de rendu: `hidentf_order/hidentf_order.c`

Sujet :

Écrivez une fonction qui renvoie 1 si toutes les lettres de s1 se cachent dans l'ordre dans s2 et 0 dans le cas inverse.

La fonction devra être prototypée de la façon suivante :

```
int hidentf_order(char *s1, char *s2);
```

Exemple :

Un appel de votre fonction avec les arguments suivants devra afficher ces résultats :

```
printf("%d",hidentf_order("abc", "abcdefghijklqsdblablabla"));
```

```
$> 1
```

```
printf("%d",hidentf_order("fgex", "tyf34gdgf;'ektufxjhgdge.;.;rtjynur6
```

```
$> 0
```

```
printf("%d",hidentf_order("“azbc”", "btarc"));
```

```
$> 0
```

Rot_42

Fichier de rendu: `rot_42/rot_42.c`

Sujet :

Écrivez une fonction qui prend une chaîne de caractères en paramètre et va incrémenter chacune de ses lettres par 42 puis l'afficher dans le terminal suivi d'un retour à la ligne.

La fonction devra être prototypée de la façon suivante :

```
void rot_42(char *str);
```

Exemple :

Un appel de votre fonction avec les arguments suivants devra afficher ces résultats :

```
rot_42("abc");
```

```
$> qrs
```

```
rot_42("abcdefghijklmnopqrstuvwxyz");
```

```
$> qrstuvwxyzabcdefghijklmnop
```

```
repeat_alpha("Vive le coding club 2022");
```

```
$> Lylu bu setydw sbkr 2022
```

N.b: Une lettre revient à 'a' lorsqu'elle dépasse 'z' et à 'A' lorsqu'elle dépasse 'Z'

Inter

Fichier de rendu: `inter/inter.c`

Sujet :

Écrivez une fonction qui prend en paramètres deux chaînes de caractères et qui affiche sans doublon les caractères communs aux deux chaînes.

L'affichage se fera dans l'ordre d'apparition dans la première chaîne.

L'affichage doit être suivi d'un retour à la ligne.

La fonction devra être prototypée de la façon suivante :

```
void inter(char *s1, char *s2);
```

Exemple :

Un appel de votre fonction avec les arguments suivants devra afficher ces résultats :

```
inter("padinton", "pagefwtdjetyiytjneytjoeyjnejejj");
```

```
$> padinto
```

```
inter("ddf6vewg64f", "gtwthgdwthdwfteewhrtag6h4ffdhsd");
```

```
$> df6ewg4
```

```
inter("rien", "cette phrase ne cache rien");
```

```
$> rien
```


Union

Fichier de rendu: `union/union.c`

Sujet :

Écrivez une fonction qui prend en paramètres deux chaînes de caractères et qui affiche sans doublon les caractères qui apparaissent dans l'une ou dans l'autre.

L'affichage se fera dans l'ordre d'apparition dans la première chaîne. L'affichage doit être suivi d'un retour à la ligne.

La fonction devra être prototypée de la façon suivante :

```
void my_union(char *s1, char *s2);
```

Exemple :

Un appel de votre fonction avec les arguments suivants devra afficher ces résultats :

```
my_union("zpadinton", "paqefwtdjetyijtjneytjoeyjnejejj");
```

```
$> zpadintoqefwjy
```

```
my_union("ddf6vewg64f", "gtwthgdwthdwfteewhrtag6h4ffdhsd");
```

```
$> df6vewg4thras
```

```
my_union("rien", "cette phrase ne cache rien");
```

```
$> rienct phas
```

Last_word

fichier de rendu : `last_word/last_word.c`

Sujet :

Écrivez une fonction qui prend une chaîne de caractères en paramètre, affiche le dernier mot de cette chaîne, suivi d'un '\n'.

On appelle "mot" une portion de chaîne de caractères délimitée soit par espaces et/ou des tabulations, soit par le début / la fin de la chaîne

La fonction devra être prototypée de la façon suivante :

```
void last_word(char *str);
```

Exemple :

Un appel de votre fonction avec les arguments suivants devra afficher ces résultats :

```
last_word("EPITECH C'EST PLUTOT COOL QUAND MEME");
```

```
$> MEME
```

```
last_word("Bientot      vous          serez      des      tek      1
```

```
$> 1
```

```
last_word("      lorem,ipsum      ");
```

```
$> lorem,ipsum
```

```
## R_capitalize
```

```
> **Fichier de rendu:** `r_capitalize/r_capitalize.c`
```

```
### Sujet :
```

Écrivez une fonction reproduisant le comportement de la fonction capit

La fonction prendra 2 paramètres, la chaîne à modifier et un entier dé

- Si order vaut 0 vous devez appliquer la fonction capitalize de javas

- Si order vaut 1 vous devez appliquer l'inverse de la fonction capita

Si order ne vaut ni 0 ni 1 vous devez simplement afficher un \n

Elle sera prototypé de la façon suivante :

```
```\n
void r_capitalize(char *str, int order);
```

## Exemple :

Un appel de votre fonction avec les arguments suivants devra afficher ces résultats :

```
r_capitalize("EPITECH C'EST PLUTOT COOL QUAND MEME !", 0);
```

```
$> Epitech C'est Plutot Cool Quand Meme !
```

```
r_capitalize("Epitech C'est Plutot Cool Quand Meme !", 1);
```

```
$> epitechH c'esT plutoT cool quand memE !
```

# Vous y êtes presque !

Toutes mes félicitations pour être arrivé jusqu'ici !

Que vous ayez tout réussi, skip un exercice sur 2 ou bien qu'il soit 9h et que tu sois entrain de lire tout le pdf par simple curiosité tu es le ou la bienvenue ici !

Il ne reste plus que 2 exercices, Au bout !

Bon courage pour la suite jeune Tek 0 !

# Pgcd

Fichier de rendu: pgcd/pgcd.c

## Sujet :

Écrivez une fonction qui renvoie le plus grand dénominateur commun entre 2 nombres.

La fonction devra être prototypée de la façon suivante :

```
int pgcd(int nb1, int nb2);
```

## Exemple :

Un appel de votre fonction avec les arguments suivants devra afficher ces résultats :

```
printf("%d",pgcd(42, 10))
```

```
$> 2
```

---

```
printf("%d",pgcd(14, 77))
```

```
$> 7
```

---

```
printf("%d",pgcd(17, 3))
```

```
$>1
```

# Fprime

Fichier de rendu: fprime/fprime.c

## Sujet :

Écrivez une fonction qui prend en paramètre un entier strictement positif, et qui affiche sa décomposition en facteurs premiers dans le terminal, suivie d'un '\n'.

Les facteurs doivent être affichés dans l'ordre croissant et séparés par des '\*', de telle sorte que l'expression affichée donne le bon résultat.

La fonction devra être prototypée de la façon suivante :

```
void fprime(int nb);
```

## Exemple :

Un appel de votre fonction avec les arguments suivants devra afficher ces résultats :

```
fprime(6)
```

```
$>2*3
```

```
fprime(225225)
```

```
$> 3*3*5*5*7*11*13
```

```
fprime(8333325)
```

```
$> 3*3*5*5*7*11*13*37
```

```
fprime(-1)
```

```
$> -1
```